***Listing of the Claims*:**

1. (Previously Presented) A system for non-intrusively monitoring an application, comprising:

at least one application stored on a computer-readable medium, the at least one application creates a shared memory area and stores application values in the shared memory area;

a first module stored on a computer-readable medium that shares and attaches to the shared memory area that is used by the at least one application during real-time operation, the first module reads application values from the shared memory area that have been stored in the shared memory area by the at least one application during real-time operation;

a second module stored on a computer-readable medium in communication with the first module that requests the first module to read the application values, the second module receives the application values from the first module; and

a third module stored on a computer-readable medium in communication with the second module that displays the application values.

2. (Previously Presented) The system of Claim 1, wherein the shared memory area is further defined as a shared memory of the application.

3. (Previously Presented) The system of Claim 1, wherein the first module is further configured to attach to the shared memory area used by the at least one application to read the application values.

4. (Previously Presented) The system of Claim 1, wherein the application values are further defined as at least one application variable and a value for the at least one application variable.

5. (Previously Presented) The system of Claim 1, wherein the first module is further configured to communicate the application values to the second module in hypertext markup language format.

6. (Original) The system of Claim 1, wherein the third module is further defined as a graphical user interface.

7. (Previously Presented) The system of Claim 6, wherein the graphical user interface is further configured to receive an input identifying the application values to be read and configured to request the application values identified to the first module, via the second module, and wherein the first module is configured to read the requested application values data from the shared memory area and return the application variables to the graphical user interface, via the second module.

8. (Previously Presented) The system of Claim 6, wherein the graphical user interface is further configured to receive an input identifying requested application values to be displayed.

9. (Previously Presented) The system of Claim 1, wherein the first module is further configured as a socket server and wherein the second module is further configured as a socket client such that the first and second modules communicate via a socket connection.

10. (Previously Presented) The system of Claim 1, wherein the first module reads application values stored in the shared memory area by the at least one application while the at least one application is running.

11. (Previously Presented) The system of Claim 10, wherein the first module reads application values stored in the memory area by the at least one application without interfering with the operation of the at least one application.

12. (Previously Presented) A method of non-intrusively monitoring operation of an application, comprising:

running an application in a real-time manner;

creating a memory area;

generating, by the application, application values during operation of the application;

writing, by the application, the application values in the memory area during the operation of the application;

reading, by a monitor, the memory area used by the application to obtain the application values, wherein at least one of the application values is not output by the application;

and

displaying the application values read from the memory area.

13. (Previously Presented) The method of Claim 12, further comprising:

requesting, by a client, application values from the monitor; and

communicating the application variables from the monitor to the client.

14. (Previously Presented) The method of Claim 13, further comprising:

requesting application values;

running a plurality of applications in a real-time manner;

generating application values stored in one or more memory areas during operation of the plurality of applications;

reading the one or more memory areas used by the plurality of applications to obtain the application values; and

displaying the requested application values.

15. (Previously Presented) The method of Claim 14, wherein the memory area is further defined as a block of memory and wherein the monitor reads at least some of the application variables stored in the block of memory.

16. (Previously Presented) The method of Claim 13, further comprising providing a memory manager and wherein the monitor registers with the memory manager to obtain a location of the memory area used by the application to store the application values.

17. (Original) The method of Claim 13, further comprising:

generating new application values by the application stored in the memory area, at least one of the new application values defined as a new value for a variable of the application;

requesting, by the client, that the monitor re-read the application values stored in the memory area;

re-reading, by the monitor, the memory area to obtain the new application values.

18. (Previously Presented) The method of Claim 17, wherein the monitor reads the application values while the application is running.

19. (Previously Presented) The method of Claim 13, wherein the monitor is configured as a socket server and wherein the client is configured as a socket client such that the communication between the monitor and client is via a socket connection.

20. (Original) The method of Claim 12, wherein the application values are further defined as a variable of the application and a value of the variable.

21. (Previously Presented) A system for non-intrusively monitoring variables during operation of an application, comprising:

a compile listing stored on a computer-readable medium having an address map with an offset associated with each of a plurality of variables of an application; and

a module stored on a computer-readable medium that performs reading of the compile listing and obtaining the offset of at least one of the plurality of variables of the application, the module performs attaching to an address space used by the application during real-time operation to obtain a value for one or more of the plurality of variables written to the address space by the application during the real-time operation of the application using the offset.

22. (Previously Presented) The system of Claim 21, wherein the module is further configured to read the compile listing and convert at least one of the plurality of variables to the associated offset.

23. (Previously Presented) The system of Claim 21, wherein the module is further configured to search the compile listing and display the plurality of variables of the application for selection by a user.

24. (Original) The system of Claim 23, wherein the module is responsive to selection by the user of one of the plurality of variables to obtain the value for the selected one of the plurality of variables using the offset to locate the value of the variable in the address space.

25. (Previously Presented) The system of Claim 24, wherein the module is further configured to display the selected one of the plurality of variables.

26. (Original) The system of Claim 21, wherein the address space is further defined as a memory space and wherein the module attaches, using a socket layer, to the memory space used by the application.

27. (Original) The system of Claim 26, wherein the module attaches, using the offset, to the memory space used by the application via an operating system service.

28. (Previously Presented) The system of Claim 21, wherein the monitor is further configured, using the compile listing, to query the address map for one or more of the plurality of variables of the application.

29. (Original) The system of Claim 21, wherein the module is further defined as a subtask of the operating system.

30. (Previously Presented) The system of Claim 21, wherein the module is further configured to attach to the memory space where the application is operating and overwrite the value for one or more of the plurality of variables using the offset.

31. (Previously Presented) The system of Claim 21, wherein the module comprises:

a reader component configured to perform reading the compile listing and further configured to perform converting at least one of the plurality of variables of the application to the associated offset; and

a search component that performs receiving the associated offset of the at least one of the plurality of variables from the reader component, the search component configured to perform attaching to the application and further operable to locate the value of the at least one of the plurality of variables using the offset.

32. (Previously Presented) The system of Claim 21, further comprising a display component operably coupled to the module to perform receiving the value for the one or more of the plurality of variables, the display component configured to perform displaying the value.

33. (Previously Presented) The system of Claim 32, wherein the display component is configured to employ the value to display a heartbeat.

34. (Previously Presented) The system of Claim 32, wherein the display component is configured to employ the value to display as a percentage complete.

35. (Previously Presented) A system for non-intrusively monitoring COBOL application values, the system comprising:

a COBOL program stored on a computer-readable medium that creates a shared memory area through a technical layer, generates program values, and stores the program values in the shared memory area during real-time operation of the COBOL program; and

a COBOL monitor module stored on a computer-readable medium that shares the shared memory area with the COBOL program through the technical layer, and the COBOL monitor module reads the program values stored in the shared memory area by the COBOL program during real-time operation of the COBOL program.

36. (Previously Presented) The system of Claim 35, further comprising:

a second COBOL program configured to generate second program values and store the program values in the shared memory area during real-time operation of the second COBOL program, and wherein the COBOL monitor module is further configured to read the second program values stored in the shared memory area by the second COBOL program.

37. (Previously Presented) The system of Claim 35, further comprising:

a second memory area; and

a second COBOL program configured to generate second program values and store the program values in the second memory area during real-time operation of the second COBOL program, and wherein the COBOL monitor module is further configured to read the second program values stored in the second memory area by the second COBOL program.

38. (Previously Presented) The system of Claim 35, further comprising:

a user interface configured to monitor and display the application values; and

a client application in communication with the user interface and the COBOL monitor module, the client application configured to request the program variables of the COBOL program from the COBOL monitor module and provide the program variables to the user interface for display via the user interface responsive to a request from the user interface.